Heikki Siikavirta

RASPBERRY PI HIUKKASANTURINA

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Joulukuu 2017

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Joulukuu 2017	**Tekijä/tekijät** Heikki Siikavirta
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn nimi Raspberry Pi hiukkassaturina		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 34
Työelämäohjaaja		

Opinnäytetyön aiheena oli Raspberry Pi-alustalle tehty hiukkasmittari. Työ tehtiin Centria-ammattikorkeakoulun sovelletun elektroniikan laboratorion käyttöön. Työn tavoitteena oli selvittää anturin DN7C2CA006 toimintamahdollisuuksia, kuten kuinka luotettava anturi on. Anturilta saadut hiukkasarvot tallennettiin tietokantaan mahdollista työstöä varten.

Alussa käsitellään teoriaosuutta, jonka jälkeen paneudutaan käytäntöosuuteen.

Työn tuloksena syntyi DN7C2CA006 hiukkasanturikokonaisuus, mutta arvojen oikeellisuudesta olen hieman skeptinen. Tulokset tallentuvat tietokantaan aikaleiman kanssa.

Asiasanat
Hiukkasmittaus, Python, Raspberry Pi.

ABSTRACT

Centria University of Applied Sciences Ylivieska	Date December 2017	Author Heikki Siikavirta
Degree programme Media technology		
Name of thesis Rasberry Pi as a dust sensor		
Instructor Hannu Puomio		**Pages** 34
Supervisor		

The subject of this thesis was Rasberry Pi as a dust sensor. The work was done for Centria university of applied sciences' lab of applied electronics. The goal was to explore the working capabilities of the DN7C2CA006 sensor. The data from the sensor was saved into database.

The first three chapters are theoretical and the rest are practical. The five chapters are: Introduction, aspects of particles, Rasberry Pi and the devices connected to it, usage of Rasberry Pi as a dust sensor and finally comparing the results and thinking about the concept as a whole .

The result was a system that reads dust values and saves them to database. Comparing the end results with data from other sensors raised questions whether the sensor is accurate or not.

Key words
Dust sensor, Rasberry Pi, Python

KÄSITTEIDEN MÄÄRITTELY

Aerosoli

On kaasun ja siinä leijuvien kiinteiden tai nestemäisten hiukkasten seos.

PWM

PWM, eli Pulse-Width Modulation tai Pulssinleveysmodulaatio. Tämän tyyppinen modulointitapa säätelee kuormaan menevää jännitettä muuttamalla pulssisuhdetta. Tämä tapahtuu niin, että lähtösignaalin keskiarvo yhden värähtelyjakson ajalta laskettuna on sama kuin modulointisignaalin arvo.

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

KUVAT

TAULUKOT

1 JOHDANTO

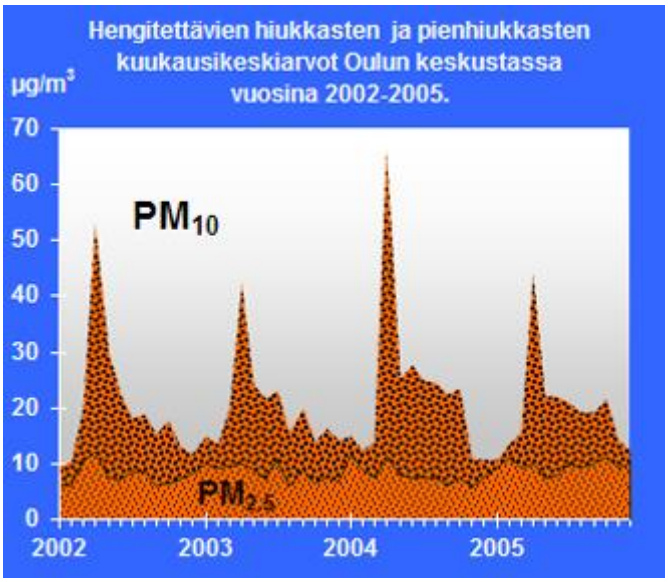
Pölyhiukkaset aiheauttavat ympäristössä erilaisia haittoja ihmisille ja laitteille. Varsinkin ihmisille pienhiukkaset aiheuttavat terveydellisiä haittoja. Hengitettävä pöly sekä pienhiukkaset aiheuttavat mm. keuhkosairauksia ja limakalvoärsytystä. Laitteille taas kerääntynyt pöly aiheuttaa toimintatehon heikentymistä sekä voi aiheuttaa sähkölaitteille myös oikosulkuja ja tulipalovaaran ympäristöön.

Työn tavoitteena oli mitata hiukkasten määrää DN7C2CA006-sensorin avulla ja tallettaa tiedot tulevaa työstöä varten johonkin tiedostoformaattiin, kuten tietokantaan. Ohjelmointikieleksi valittiin Python-kieli sen laajan levinneisyyden ja yksinkertaisuuden takia.

Alussa paneudutaan lyhyesti hiukkasten luonteeseen. Seuraavassa pääluvussa perehdytään hiukkasanturin teoriaosuuteen. Seuraavaksi perehdytään käytännön osuuteen, ja selvennetään sitä, miten Rasberry Pi:tä ja oheislaitetta käytettiin, jotta saavutettiin haluttu lopputulos. Lopuksi pohditaan työn ongelmia ja onnistumisia sekä työn tuloksia, joita verrataan muiden antureiden tuloksiin.

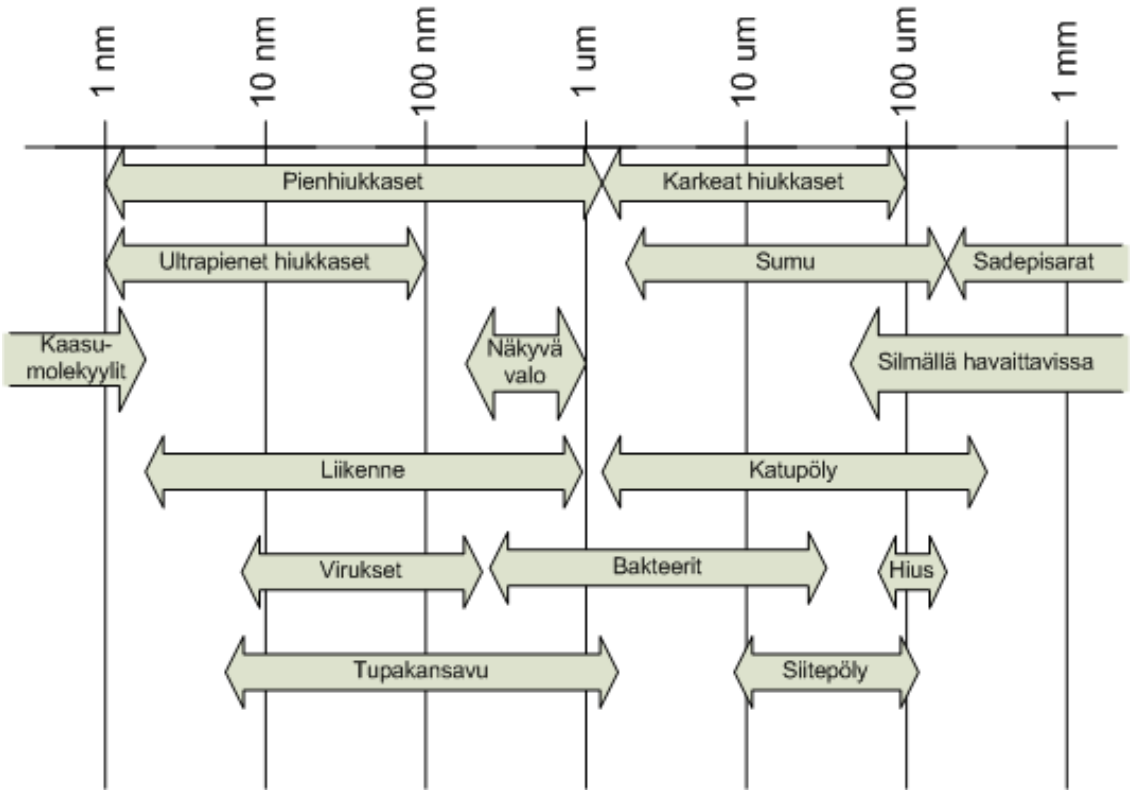
2 HIUKKASTEN KOKOLUOKITUS

Hiukkasia, jotka ovat alle 2.5 mikrometrin (µm) kokoisia kutsutaan pienhiukkasiksi, ja ne voidaan myös merkitä lyhenteellä PM2.5. Nämä pienhiukkaset ovat osa ihmisten jokapäiväistä hengitettävää ilmaa. Esimerkkinä on kuvassa 1 Oulun keskustassa mitattuja arvoja. Pienemmät hiukkaset tunkeutuvat hengitysilman mukana syvemmällä hengitysteihin aiheuttaen ihmisille erilaisia terveydellisiä haittoja. Pienhiukkasia syntyy ilmakehään muun muassa polttoaineiden palamisessa sekä puita polttaessa, esimerkiksi vanhemmille suomalaisille tutusta kaskeamisesta. (Ilmanlaatu, 2017.)



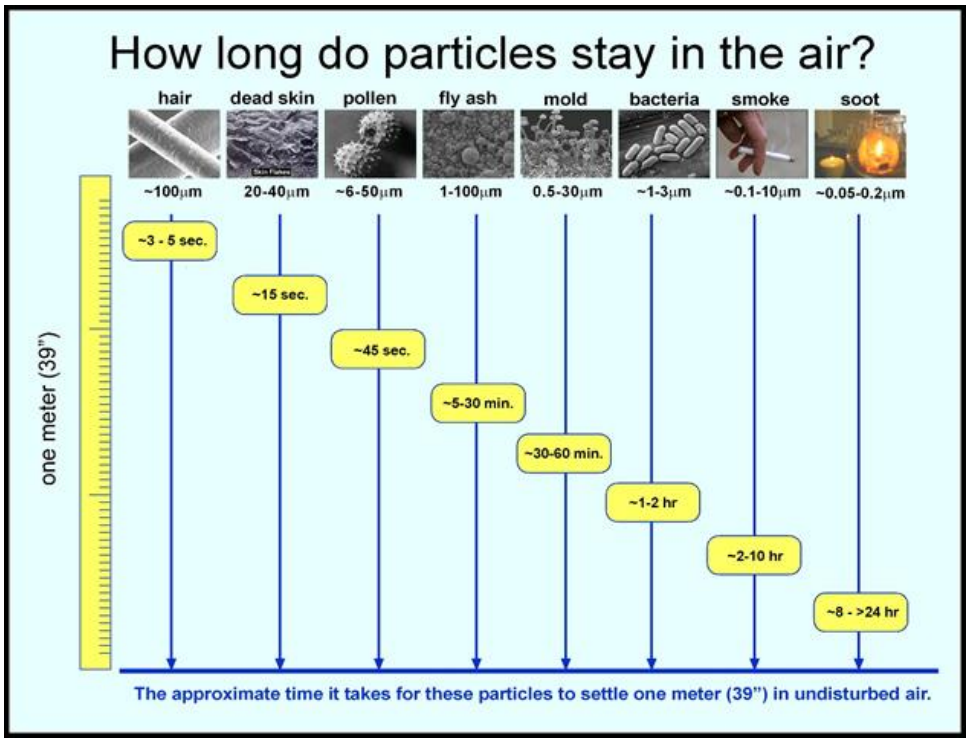
KUVA 1. Hiukkasarvoja Oulusta (Ilmanlaatuportaali, 2017)

2.1 Lyhyesti hiukkasten koko ja muoto



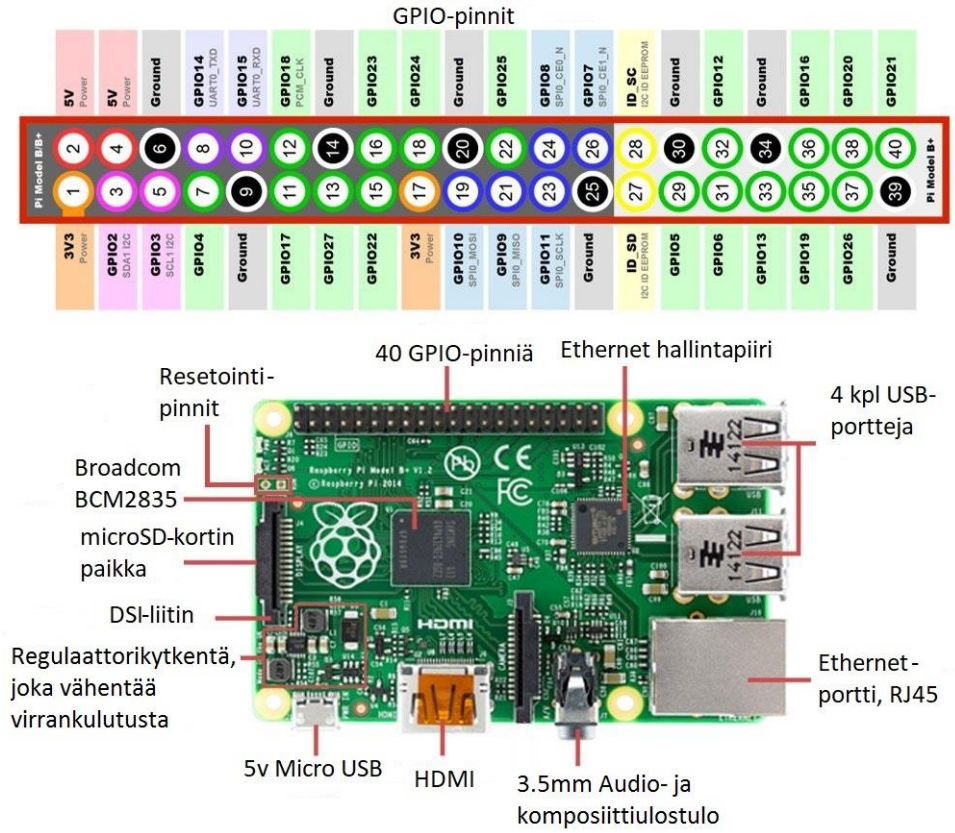
KUVA 2. Hiukkasten kokoluokat (Hiukkastieto, 2017)

Hiukkasia käsiteltäessä on hyvä keskustella niiden suuruusluokista. Hiukkaskoon mittayksikkö on mikrometri(µm). Hiukkasen koon määrittää yleensä sen halkaisija. Aerosolihiukkasten kokoluokaksi on määritelty 1 nm – 100 µm:n halkaisija. Huomataan, että hiukkasten kokoluokka voi vaihtua moninkertaisesti siinä missä pienimmät hiukkaset ovat vain muutamista molekyyleistä rakentuvia ryppäitä, kun taas isommat hiukkaset taas ovat jo ihmissilmin havaittavissa olevia, kuten pölyhiukkaset. Hiukkasten kokoalue rajataan ylä- ja alarajaan. Kuvassa 2 on esimerkkejä aerosolihiukkasista ja niiden halkaisijoista. Ylärajana toimii hiukkasten viipymisaika ilmakehässä ja alarajan määrää hiukkasia muodostavien molekyylien koko. Kuvassa 3 on hiukkasten viipymisaikoja ilmassa. (Hiukkastieto, 2017.)



KUVA 3. Hiukkasten viipymisaika ilmassa (Climate Works, 2017)

3 RASPERRY PI JA KÄYTETYT ANTURIT



KUVA 4. Rasperry Pi (mukaillen Jameco, 2017)

Rasberry Pi Foundation on luonut Rrasperry Pi -pientietokoneen, lyhyemmin RasPi. Opinnäytteessä käytettiin **RasPi 3 mode B** versiota. (Rasberry Pi, 2017) Kuva 4 ja taulukko 1 sisältävät hyödyllisiä tietoja.

	Raspberry Pi 1 (Model B+)	Raspberry Pi 2 (Model B)	Raspberry Pi Zero	Raspberry Pi 3 (Model B v1.2)
Julkaisupäivä:	14.7.2014	2.2.2015	26.11.2015	29.02.2016
Myyntihinta:	25 USD	35 USD	5 USD	35 USD
Prosessori:	700 MHz, Yksiytiminen, ARM11,	900 MHz, Moniydinprosessori,ARM Cortex-A7,	1 GHz, Yksiytiminen, ARM11,	1.2 GHz, Moniydinprosessori, ARM Cortex-A53,

	32-bittinen, ARMv6	32-bittinen, ARMv7-A	32-bittinen, ARMv6	64-bittinen, ARMv8-A
Näytönohjain:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p, h.264, 250MHz			BroadcomVideoCore IV, OpenGL ES 2.0, 1080p, h.264, 400MHz
Piirisarja:	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2836 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2837 (CPU, GPU, DSP, SDRAM)
Muisti:	512 Mt (jaettu näytönohjaimen kanssa	1 Gt (jaettu näytönohjaimen kanssa)	512 Mt (jaettu näytönohjaimen kanssa)	1 Gt (jaettu näytönohjaimen kanssa)
USB 2.0 portit:	4 kpl		2 kpl (microUSB)	4 kpl
Videoulosulot:	HDMI (rev 1.4), komposiitti RCA (PAL & NTSC), Model A+/B+ jakkiliitin korvaa RCA:n		miniHDMI, GPIO (komposiitti)	HDMI (rev 1.4), komposiitti RCA (PAL & NTSC), Model A+/B+ jakkiliitin korvaa RCA:n
Ääni:	HDMI, 3,5 mm jakki		miniHDMI, GPIO	HDMI, 3,5 mm jakki
Massamuisti:	SD / MMC / SDIO-muistikortti, Model A+/B+ vain			Secure Digital
Verkkosovitin:	10/100 Ethernet (RJ45)	10/100 Ethernet (RJ45) (USB-Ethernet-silta)	Ei ole	10/100 Ethernet (RJ45) (USB-Ethernet-silta)
Langattomat yhteydet:	Ei ole			**802.11n ja Bluetooth 4.1**
Muut liittimet:	40-pinninen GPIO-liitin			
Tehokulutus:	310 mA, 1,55 W (0,31 amps)	420 mA, 2,1 W (0,42 amps)	~160 mA (0.8 W)	580 mA, 2,9 W (0,58 amps)
Virtalähde:	5 V MicroUSB			

Koko:	85,60 × 53,98 × 17 mm	65 mm × 30 mm × 5 mm	85,60 × 53,98 × 17 mm
Paino:	45 g	9 g	**9 g**
Käyttöjärjestelmät:	Arch Linux, Debian (Raspbian), Fedora (Pidora), RISC OS, Slackware		

TAULUKKO 1. Rasberry Pi eri mallit (Wikipedia, 2017)

3.1 Hiukkasanturi DN7C3CA006

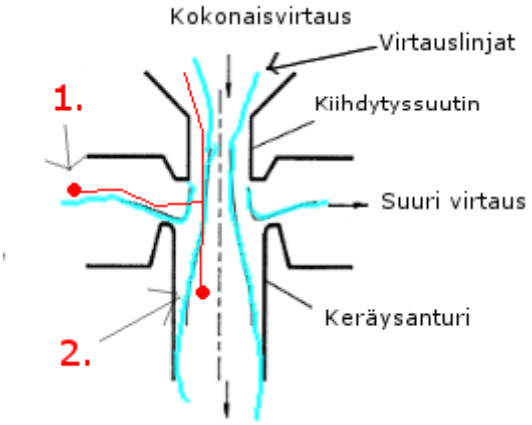
Opinnäytteessä käytettiin Sharp-yhtiön valmistamaa pölynmittaussensoria DN7C2CA006. Anturissa on sisäänrakennettu moduuli, joka mahdollistaa pienten partikkelien tunnistuksen. Anturi käyttää tuulettimelta tulevaa ilmanpainetta apuna partikkelien erotteluun ja tunnistamiseen. Anturi pystyy tunnistamaan jopa PM2.5-kokoisia ja pienempiä partikkeleita. Tunnistustarkkuus on 1V±15%/(100µg/m³). Anturissa on sisäänrakennettuna moduulina virtuaali-impaktori, jonka lävitse ohjataan tuulettimella partikkeleita. Tästä partikkeleiden tuulahduksesta voidaan erotella erikokoisia partikkeleita. (Digikey, 2015.)

KUVA 5. DN7C2CA006 Anturista (Hackster, 2016)

Partikkelien tunnistus tapahtuu sytyttämällä hetkellisesti sisäinen LED-valo. Tästä valon purskahduksesta pienet partikkelit heijastavat takaisin niihin kohdistettua valoa. Anturin sisäinen logiikka sitten muuntaa lukemat analogiseksi jännitevaihteluksi, joka ohjataan ulos Vo-pinnistä. (Hackster, 2016.)

3.1.1 Virtuaali-impaktori

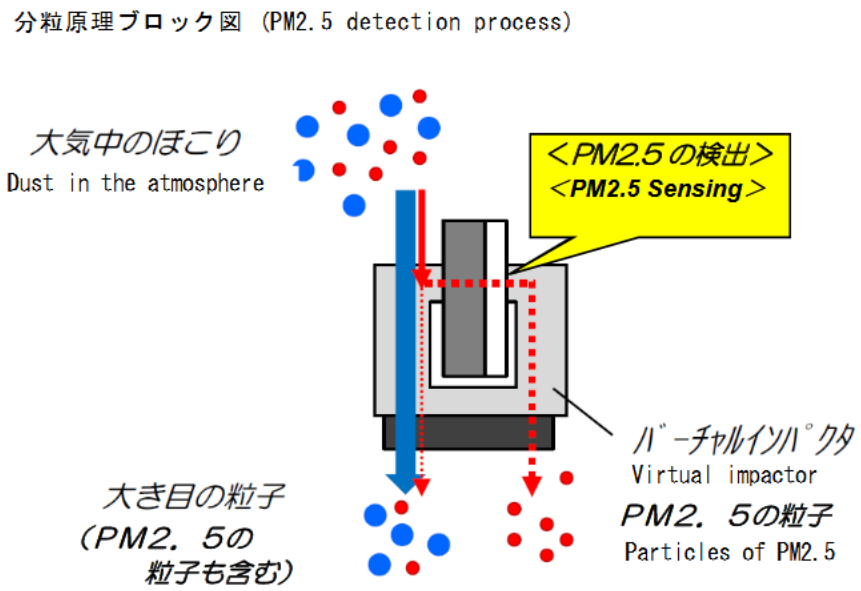
Virtuaali-impaktori on ratkaisu, jolla pystytään erotelemaan partikkeleita toisistaan kokonsa perusteella. Tämä tapahtuu käyttämällä apuna erilaisia ilmavirtoja. Mittausvaiheessa ilmavirtauksen liikesuunta muuttuu jyrkästi, jolloin määritettyä suuremmat hiukkaset eivät ehdi kääntyä virtauksen mukana. Tällä tavalla voidaan hiukkaset jakaa kahteen osaan niiden aerodynaamisen halkaisijan perusteella. Virtuaali-impaktori toimii samantyyppisesti kuin normaali impaktori, mutta keräyslevyn tilalla on pieni aukko,johon suuremmat hiukkaset seuraavat pientä ilmavirtausta. Kuviossa 6 virtuaali-impaktorin esitetään toimintamekanismi pääpiirteittäin. (Hiukkastieto, 2017.)



KUVIO 6 . Virtuaali-impaktorin toiminta (mukaillen Ensor 2011, 509)

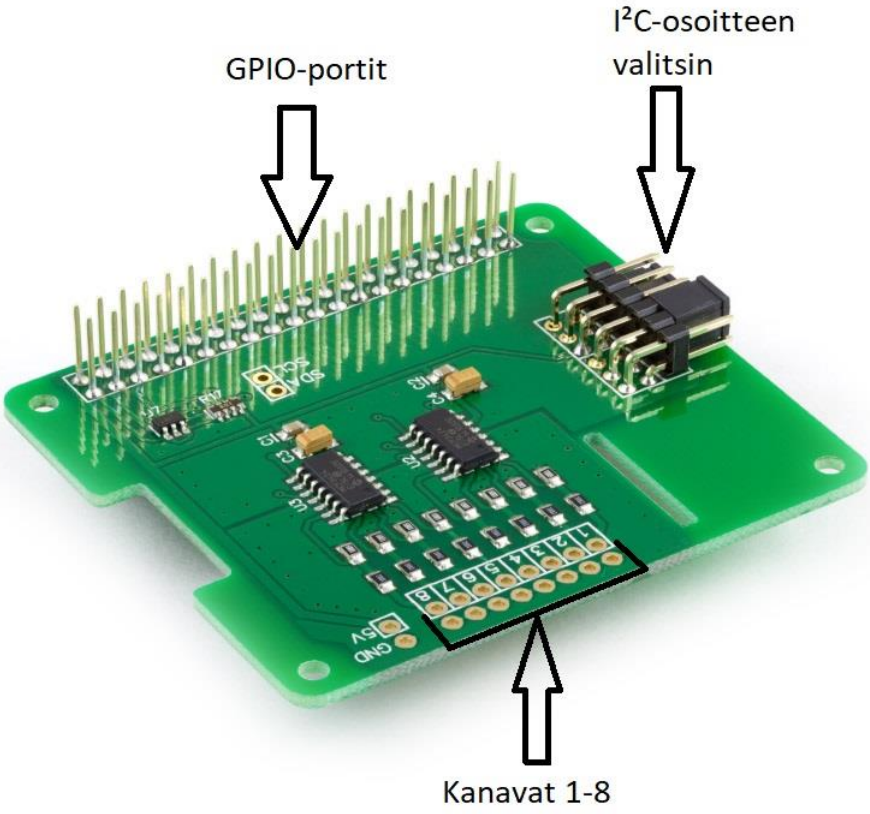
Ilmavirtaus virtaa kiihdytyssuuttimen lävitse kohti keräysanturia. Ennen sitä se kuitenkin joutuu kohtaamaan vaakasuunnassa olevan virtauksen. Tätä prosessia kutsutaan partikkelikoon jaotteluprosessiksi, ja se on oleellisin vaihe virtuaali-impaktorin toiminnassa. Partikkelit, joilla on pienempi pinta-ala, ovat virtauksien helpommin liikuteltavissa. Tämän ominaisuuden avulla pienempiä partikkeleita voidaan tehokkaasti lajitella. Kuvion 6 kohdassa 1 kuvataan partikkelia, jolla on pieni pinta-ala. Partikkeli tulee sisään kiihdytyssuuttimen kautta, jonka jälkeen partikkeliin vaikuttaa vaakasuunnassa oleva suuri virtaus, joka muuttaa partikkelin kulkurataa kääntäen partikkelia poispäin keräysanturista. (Ensor 2011, 509.)

Yhteenvetona voi siis todeta, että pienet partikkelit, joilla on vähän pinta-alaa, seuraavat virtauslinjoja ja ajelehtivat pois suuren virtauksen mukana. Suuren pinta-alan omaavat partikkelit ovat taas vaikeammin liikuteltavissa kun joutuvat kohtaamaan suuren virtauksen. Esimerkkinä kuvion 6 kohdassa 2 partikkeli omaa suuren pinta-alan. Kun tämä partikkeli tulee suuren virtauksen kohdalle, muuttaa virtaus hieman partikkelin liikerataa ja se jatkaa matkaansa virtauslinjoja pitkin kohti keräysanturia. Kuvassa 7 on esitetty valmistajan kuvaus anturin DN7C3CA006 virtuuali-impaktorista.



KUVA 7. Valmistajan kuvaus Virtaali-impaktorista (Sharp corporation 2014, 6)

3.2 ADC Pi Plus



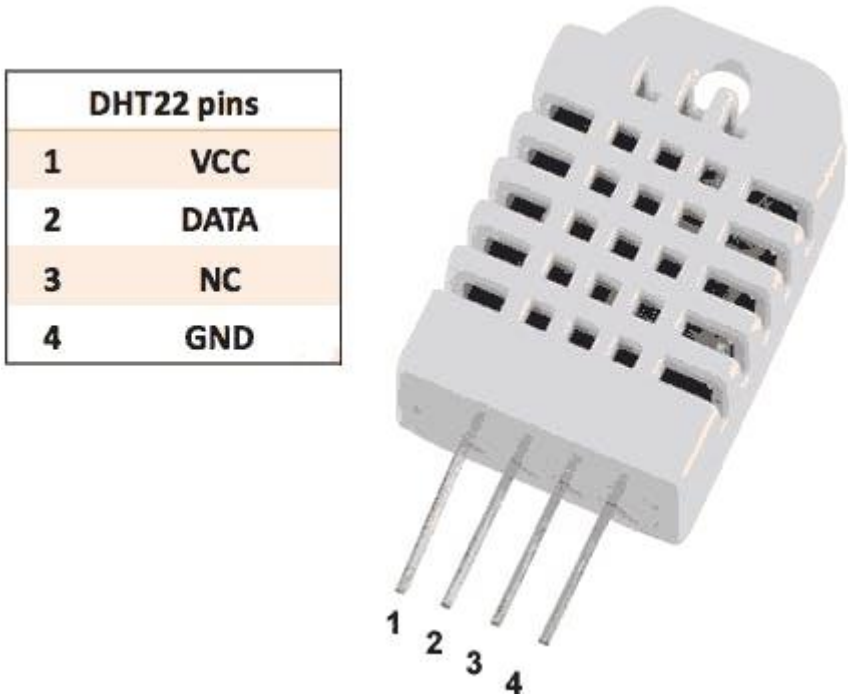
KUVA 8. ADC Pi Plus (mukaillen ABElectronics, 2017)

ADC Pi Plus on AB Electronics UK:n valmistama kahdeksankanavainen analogia-digitaalikonvertteri, eli A/D-muunnin. Kuvasta 8 nähdään ylöspäin osoittavat pinnit, jotka kiinnittyvät Rasberry Pi:n GPIO-portteihin. Kiinni ollessaan ADC Pi Plus ottaa virran suoraan Rasberry Pi:in GPIO-portista. Analogia-digitaalimuunnoksen datasuhde voi olla 3.75 (17 bittiä), 15 (15 bittiä), 60 (13 bittiä) tai 240 (11 bittiä) näytekertaa sekunnissa. Kuvassa 8 nähdään ADC Pi Plus -kortti ja sen osat, kuten esimerkiksi osoitteenvalitsin, jolla mahdollistetaan osoitteen vaihtaminen jumpperia siirtämällä. (ABElectronics, 2017.)

3.3 DTH22 lämpö ja kosteussensori

Työssä tarvittiin kosteus- ja lämpötietoa, joiden mittaamiseen valittiin DTH22-anturi. DN7C2CA006 tarvitsee kosteustietoa, jotta partikkelien laskeminen onnistuisi mahdollisemman tarkasti. Tästä enemmän luvussa 4.4.3.

DHT22 on alemman hintaluokan kosteus- ja lämpötila-anturi. Anturissa on kapasitiivinen ilmankosteusanturi ja termistori, joka nuuskii ilmanlämpötilaa. Ulostulo on digitaalimuodossa. Suurimpana haittapuolena on 2 sekunnin rajoite jokaiselle pyynnölle, eli anturilta voi antaa tietoa vain 2 sekunnin välein. (Adafruit, 2017.)



KUVA 9. DTH22 lämpö- ja kosteusensori (Arduino project hub, 2017)

Oleellisimmat DTH22:n tiedot on esitetty taulukossa 2. Lisäksi huomataan KUVASTA 9, että DATA-ulostulo on pinnissä 2. Pinniin 1 menee VCC ja pinniin 4 GND. Pinniin 3 ei kytketä mitään.

Mittausalue	Kosteus: 0-100%RH	Lämpötila: –40~125°C
Käyttöjännite	3.3 - 6v DC	
Mittausväli	2s	
Mittaustarkkuus	Kosteus: 0.1%RH	Lämpötila: 0.1°C

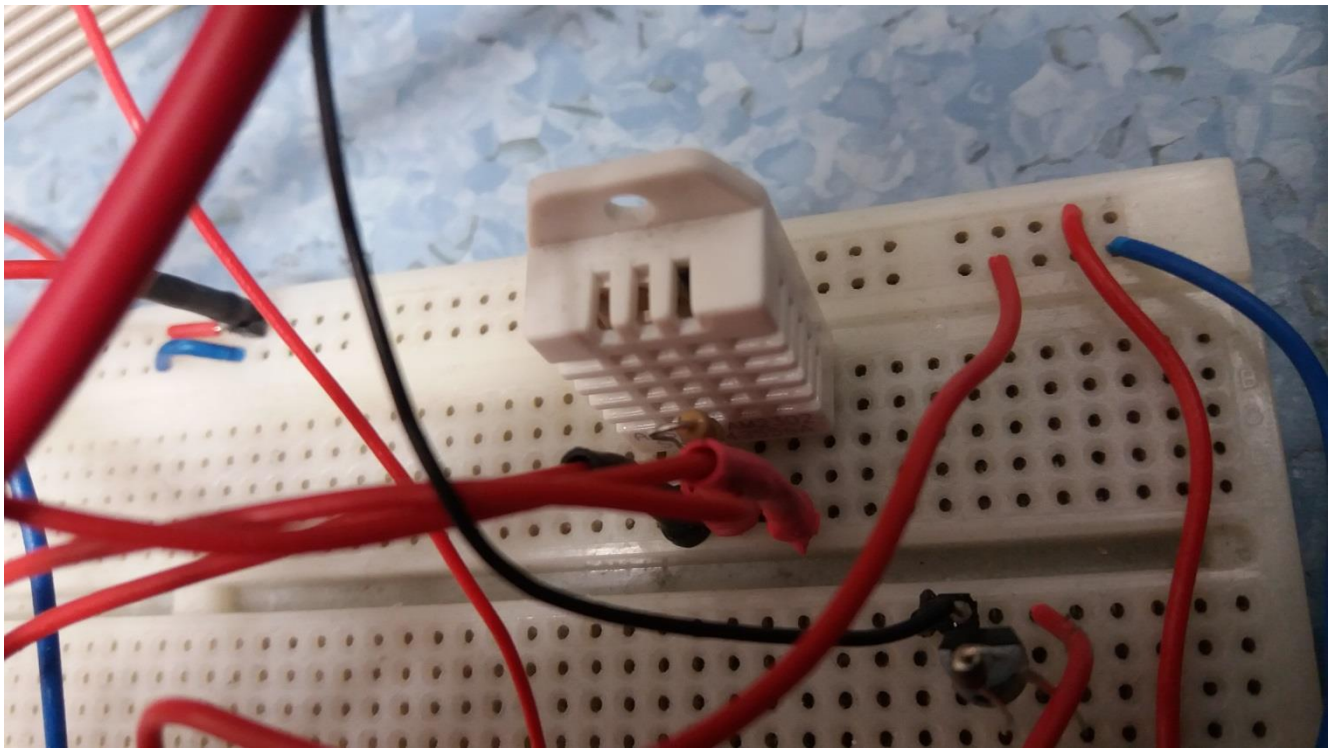
TAULUKKO 2. Muutamia oleellisia tietoja DTH22:sta (mukaillen Aosong Electronics 2017, 1-2)

4 HIUKKASANTURI RASPBERRY PI:LLÄ

Hiukkasanturi Rasberry Pi:llä muodostui itse anturin DN7C3CA006 toiminnasta ja siihen ympärille rakennetuista pienemmistä kokonaisuuksista, kuten kosteus- ja lämpötila-anturista, ADC Pi Plus – A/D-muuntimesta ja tuulettimen kontrollointikytkennästä.

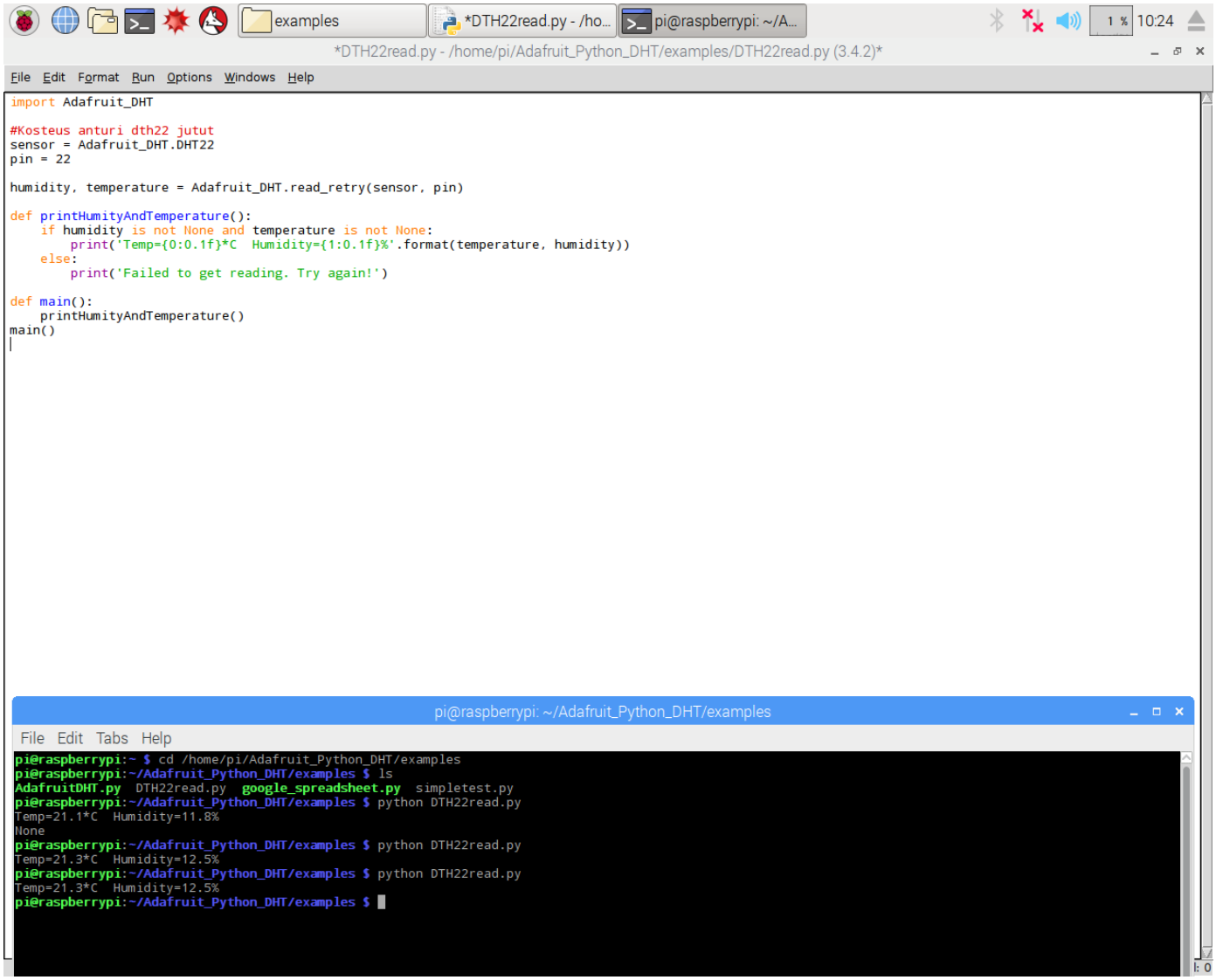
4.1 Kosteus- ja lämpötila-anturi DHT22

DTH22-anturin käyttäminen oli yksinkertaista. Aluksi piti asentaa muutama työkalu ja kirjasto, jolla anturia luetaan. Tämän jälkeen tarvitsi vain kytkeä sensori etuvastuksella, jonka jälkeen voidaan lukea dataa.



KUVA 10. DTH22 anturi kytkettynä

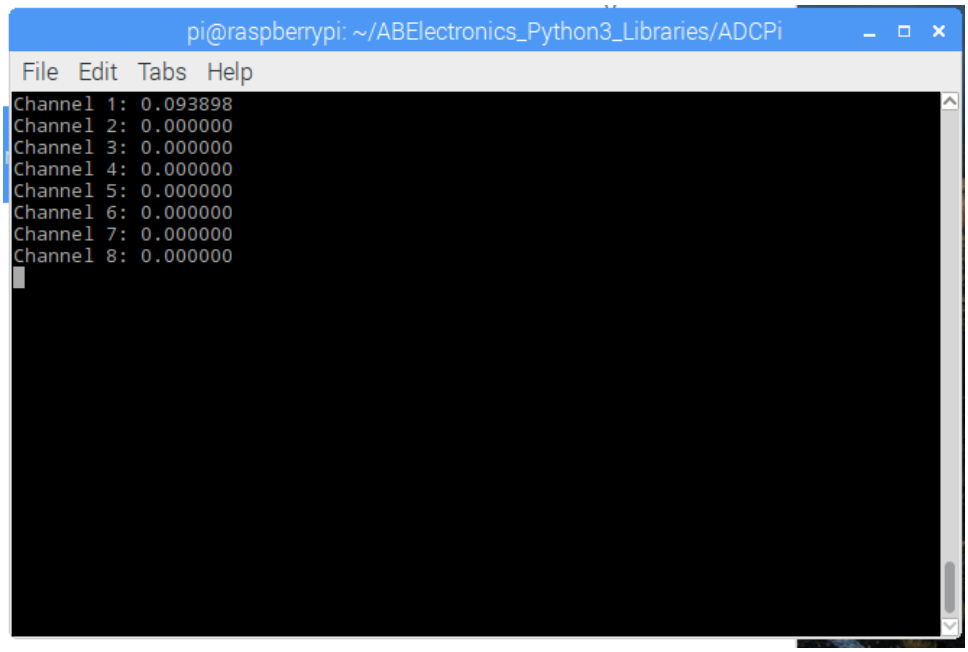
DTH22:n käyttö tarvitsee ylösvetovastuksen DATA- ja VCC-pinnien välille. Kuvasta 9 huomataan, että nämä ovat pinnit 1 ja 2. Vastukseksi valittiin 4,7 kilo ohmin ylösvetovastus. Kuvassa 11 nähdään kosteusanturin toiminta sekä siihen liittyvä koodi. Anturin luku tapahtuu Adafruit_DHT.read_retry(sensori, pinni)-metodilla. Kutsuun pitää muistaa määritellä sensoriksi kyseinen malli, eli Adafruit_DHT.DHT22 ja GPIO-portin pinni johon DHT22-sensorin DATA-pinni on kytketty, tässä tapauksessa Rasberry Pi:n GPIO-porttien pinniin 22. Kuvassa 10 näkyy anturi kytkettynä ylösvetovastuksen kanssa.



KUVA 11. DHT22 lämpö- ja kosteussensorin toiminta

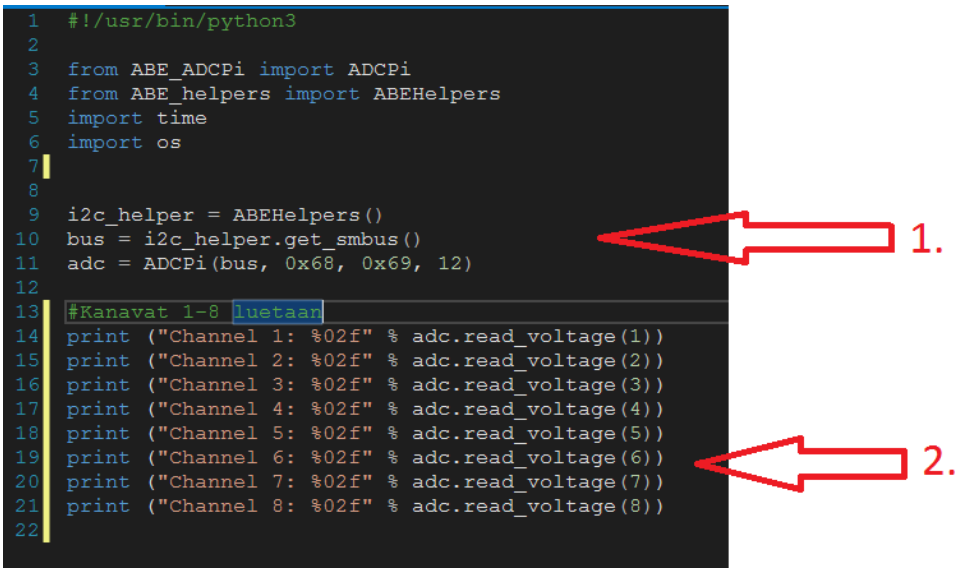
4.2 ADC Pi Plus A/D -muuntimen käyttäminen

Anturilta DN7C3CA006 tuleva tieto on analogista jännitevaihtelua, eikä Raspberry Pi 3:ssa ole itsessään sisäänrakennettua analogista sisääntuloa. Tarvittiin siis ADC-muunnin, jolla mahdollistettiin tiedon saanti Raspberrylle ymmärrytettävään muotoon. ADC Pi Plus kiinnitettiin Rasberry Pi:n GPIO-porttien päälle. Tämän jälkeen piti asentaa muutama ohjelma, joilla ADC Pi Plus onnistuu kommunikoimaan Python-koodin kanssa. Python3-kirjaston asennus onnistui ABElectornicsin git hub -palvelimelta, ja lisäksi piti sekä kytkeä päälle että asentaa i2c ja python-smbus.



KUVA 12. ADC Pi Plus arvojen lukeminen

Kuvassa 12 luetaan kanavalta 1 saatua dataa. Datan luku onnistui ABElectronics:in tarjoamalla esimerkkikoodilla, joka on kuvassa 13. Kuvassa demonstroidaan kanavien lukua palalla yksinkertaista koodia. Olennaisinta koodissa on adc- ja smbus-objektien initialisointi kohdassa 1 ja arvojen tulostus adc.read_voltage(1) -metodilla kohdassa 2.



KUVA 13. ADC Pi Plus arvojen esimerkkiluku

4.3 Datan tallentaminen tietokantaan

Mittaustulokset on hyvä tallentaa johonkin tallennusformaattiin tulevaa työstöä varten. Tulokset tallennettiin MySQL-tietokantaan yksinkertaisen python-skriptin avulla. Kuvassa 14 tietokantaskriptin toiminnot on ohjelmoitu luokaksi. Tämä luokka sisältää kolme metodia tai toisin sanoen kolme pääasiallista toimintoa(kysy): kirjautumis- (loginDatabase), partikkeliarvon tallennus- (saveParticleValue) ja yhteyden lopetus -toiminnot (closeCon)

Luokassa jokaisella metodilla on oma tehtävänsä. Kirjautumistoiminnolla kirjaudutaan MySQL-tietokantaan ja valitaan työstettävä tietokannan taulu. Tallennustoiminnolla kirjoitetaan tauluun seuraavat arvot: lukuhetken ilmankosteus, lämpötila, partikkelien pienhiukkastaso PM2.5 ja aika, jolloin tietokantaan kirjoitus tapahtui. Tämä noudattaa MySQL-tietokannan "INSERT INTO" -kyselyn alla olevaa muotoilua.

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Viimeisenä on yhteyden lopetustoiminto, jolla suljetaan yhteys turvallisesti.

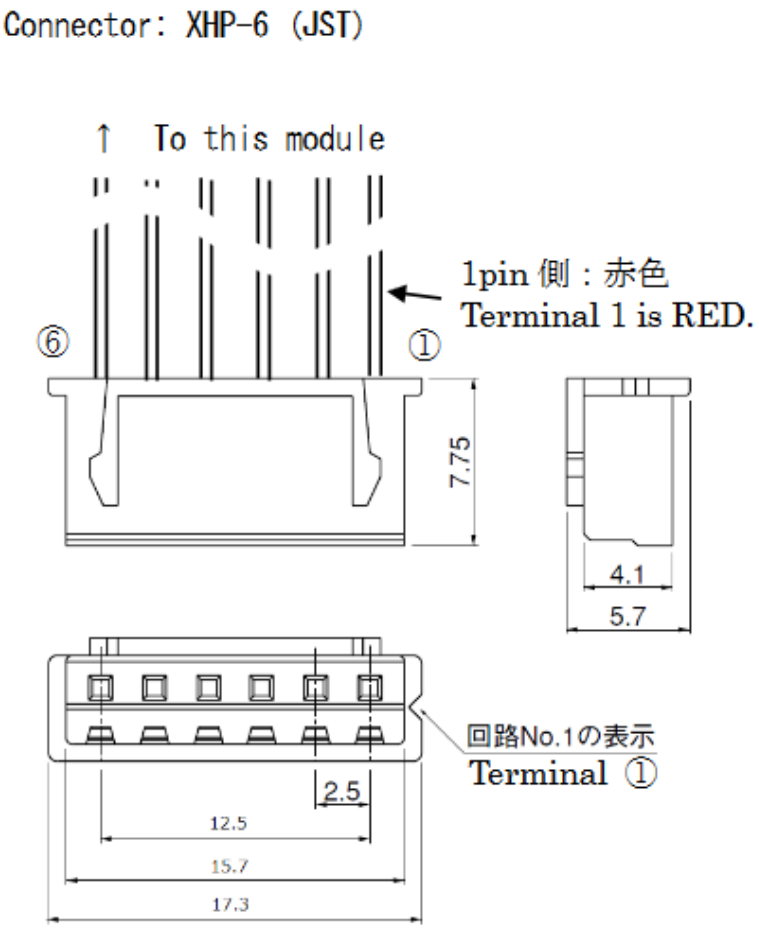
```
import sys
import MySQLdb

class DatabaseClass:

    db = None
    cur = None

    def loginDatabase(self):

        try:
            self.db = MySQLdb.connect(host="localhost",
                                      user="root",
                                      passwd="root",
                                      db="dbSensorData")
            self.cur = self.db.cursor()
        except MySQLdb.Error, e:
            self.closeCon()
            print e
            print e.args
            sys.exit(1)


    def saveParticleValue(self, kosteus, lampo, pm_2_5_taso):
        try:

            self.cur.execute(str(("""INSERT INTO TestData (kosteus, lampo, pm_2_5_taso, aika)
            VALUES ('{0}','{1}','{2}',now())""").format(kosteus, lampo, pm_2_5_taso)))
            self.db.commit()


        except Exception, e:
            print e
            print e.args
            self.db.rollback()
            self.closeCon()
            sys.exit(1)


    def closeCon(self):

        if (self.cur is not None):
            print("Curr lopetetaan")
            self.cur.close()


        if(self.db is not None):
            print("Db lopetetaan")
            self.db.close()
```

KUVA 14. Tietokantaskripti

Toiminnot on ohjelmoitu try-catch -poikkeuksenkäsittelylohkoon. Tällä lohkolla voidaan testata halutun osion, tässä tapauksessa "try"-osion toimivuus. Mikäli kyseisessä osiossa ilmenee virheitä, suoritetaan "except"-lohko, johon on ohjelmoitu yhteyden lopetus ja kyseisessä yhteydessä tehtyjen muutosten kumoaminen.

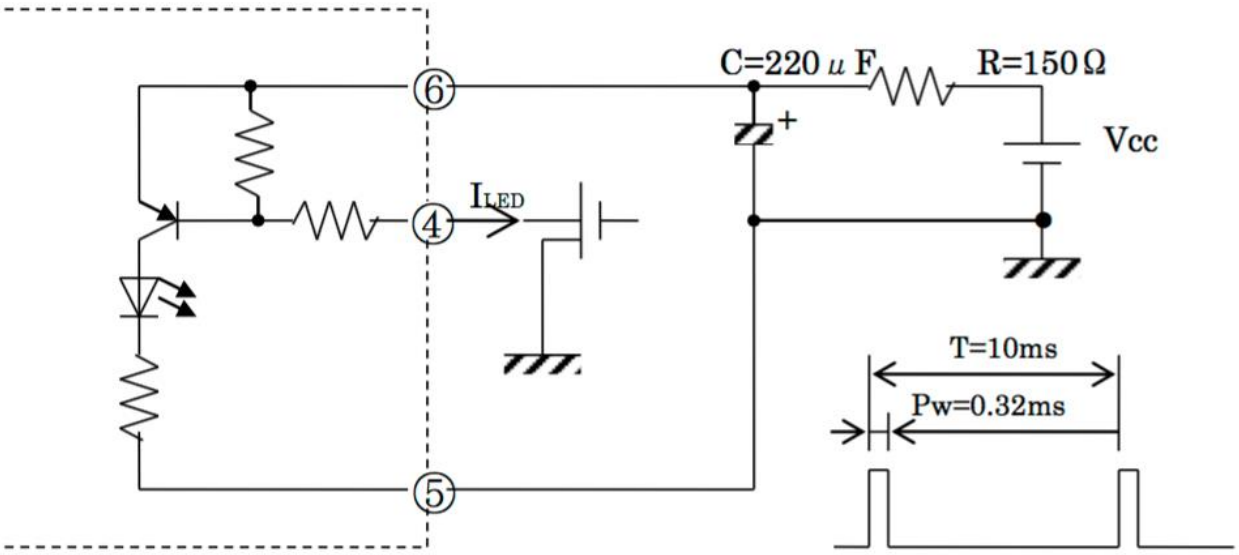
4.4 Partikkelimittari DN7C3CA006

Aluksi piti selvittää, miten kytkentä tapahtui. Tässä auttoi valmistajan tarjoama ohjekirja. Anturista tuli ulos lattakaapeli, jolla luetaan ja kontrolloidaan anturia, sekä virtajohdot tuulettimelle. Kuvassa 15 esitetään sensorilta tulevan lattakaapelin johtojen numerointia. Kuvassa 15 oikeanpuoleinen punainen kaapeli on numero 1 ja tästä eteenpäin numerointi kasvaa yhdellä aina kuvan vasemman puoleiselle johtimelle, joka on numero 6.



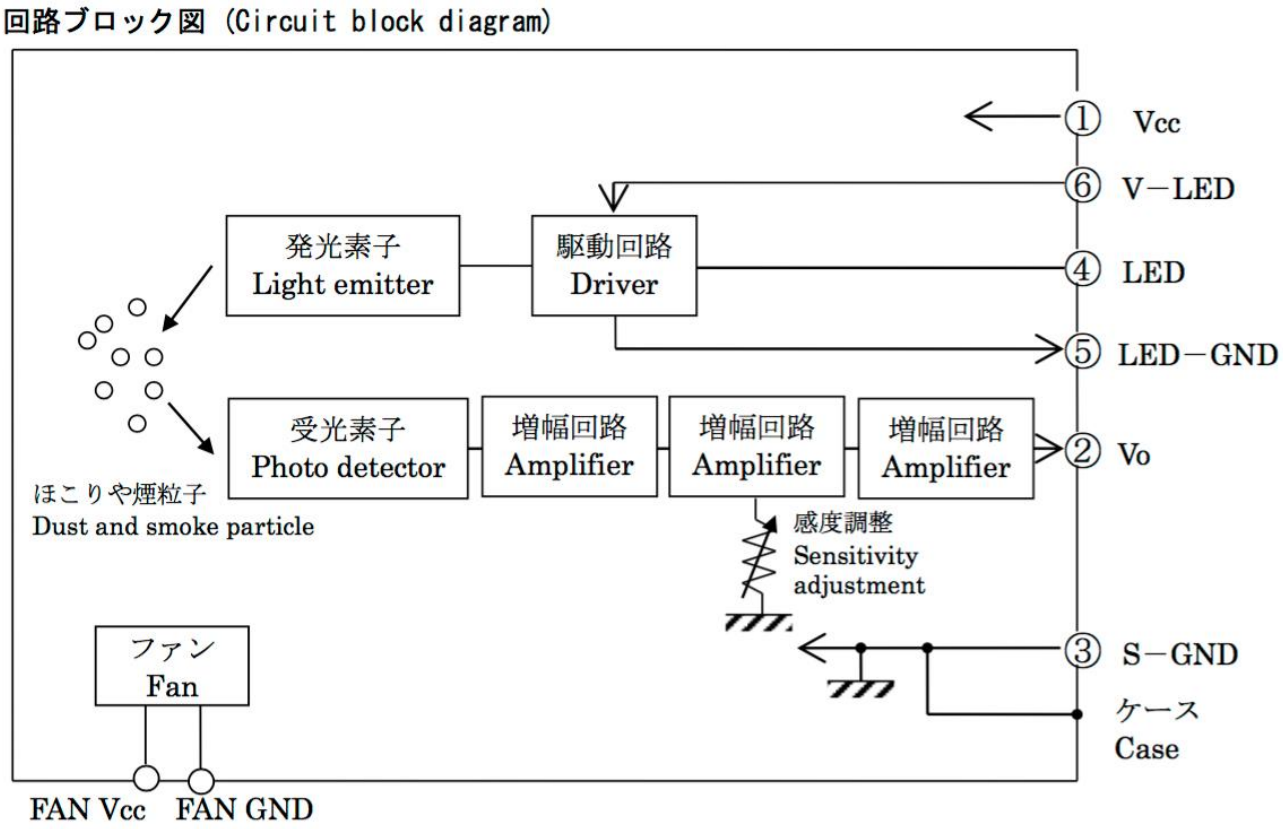
KUVA 15. DN7C3CA006:stä tulevan lattakaapelin numerointi (Sharp corporation 2014, 7)

Kytkentä ja tulkinta tapahtui kuvioiden 16 ja 17 mukaisesti. Jännite kytkettiin kuvan 17 johtimeen 1, maa johtimeen 3 ja jännitetieto tulee ulos johtimesta 2. Tuuletin menisi maahan ja jännitteeseen, mutta tuulettimesta tehtiin kontrolloitava, joten kytkentä ei ollut kuvan mukainen. Tästä enemmän luvussa 4.4.4. Kuvan 16 kohtaan 4 tulee ledille ohjaussignaali, tähän paneudutaan luvuissa 4.4.1 ja 4.4.2. Pinnit 5 ja 3 menevät maahan. (Sharp corporation 2014, 7-10.)



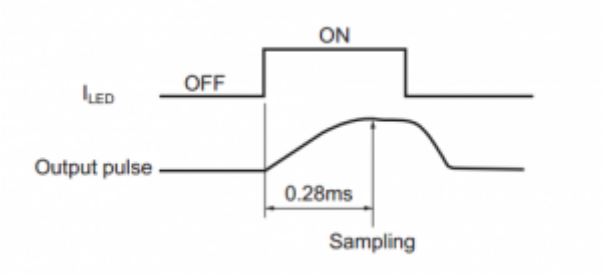
KUVA 16. Kondensaattorin ja vastuksen kytkentä sekä pulssi (Sharp corporation 2014, 10)

Lopuksi LED-ohjain tarvitsee erillisen ohjauksen, eli kuten kuvasta 16 huomataan, kohtaan 6 laitetaan 150 ohmin vastuksen kautta jännite ja sen rinnalle 220uF kondensaattori.



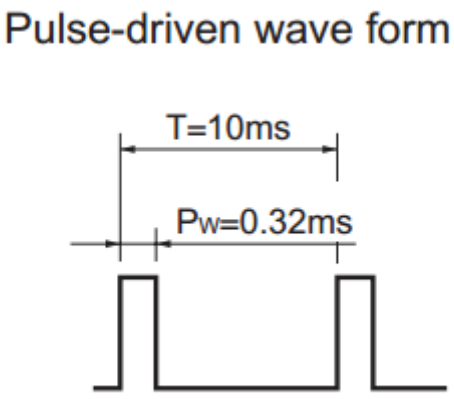
KUVA 17. DN7C3CA006 kytkentäkaavio (Sharp corporation 2014, 7)

4.4.1 Näytteenottoaika



KUVA 18. Näytteenottoaikoja (Sharp corporation 2014, 10)

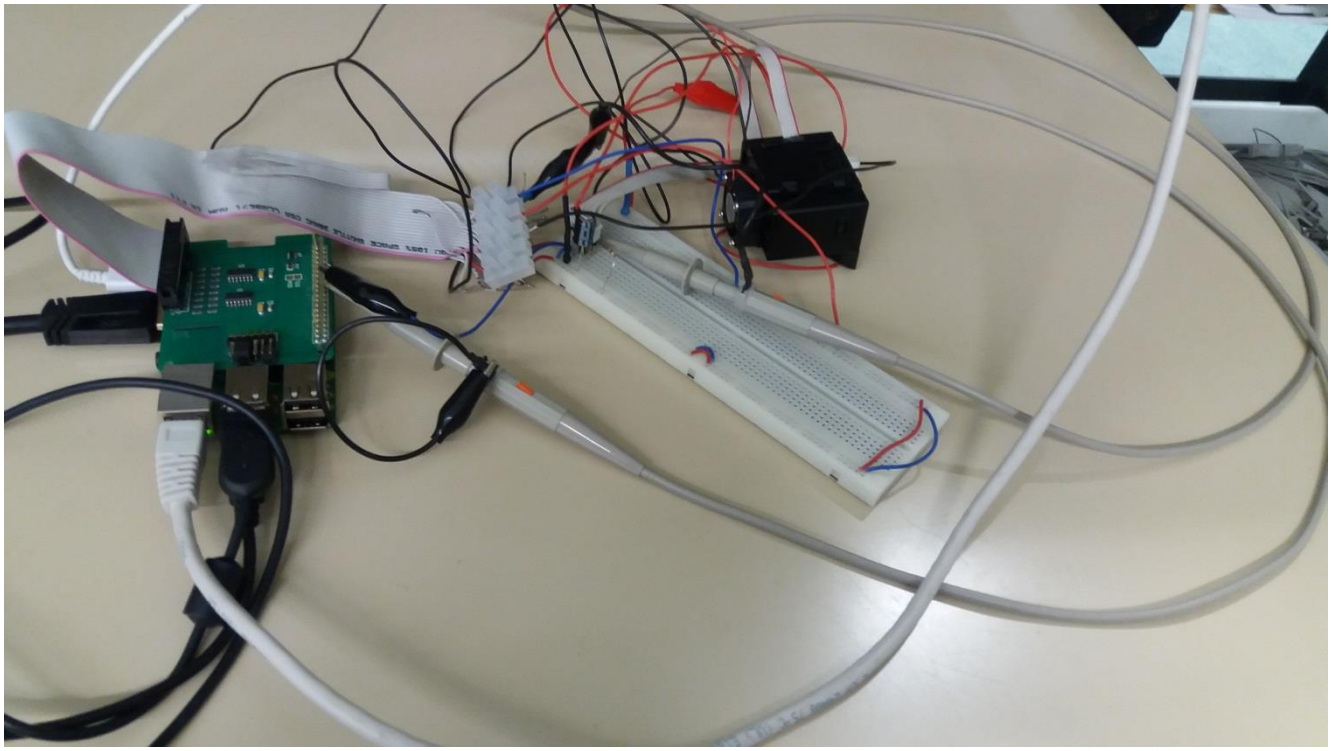
DN7C3CA006 tarvitsee ajallisesti tarkkaa ohjauspulssia, jotta partikkelin luku onnistuisi. Oli siis tarpeen tutkia anturin näytteenottoaikoja. Datasheetistä sekä kuvista 18 ja 19 huomataan, että sisäistä LED:iä tarvitsee väläyttää 10 ms:n välein. Jokaisella väläytyksellä pulssinlaajuus on 10ms. Väläytyksen jälkeen pitänee odottaa 280us, jotta saavutetaan pulssihuippu. Tästä pulssihuipusta voimme ottaa näytteen lukemalla V0-ulostuloa. Tämä ulostulo ei kuitenkaan ole vielä haluttu lopputulos. Saatuja arvoja joudutaan muuttamaan hieman, tästä enemmän luvussa 4.4.3. (Sharp corporation 2014, 10.)



KUVA 19. DN7C3CA006 toimintakaavio (mukaillen Sharp corporation 2014, 10)

LED:ille syötettävä ohjauspulssi on PWM-muotoa. Tähän aiheeseen paneudutaan seuraavassa luvussa.

4.4.2 Pulssin luominen ja vastauksen lukeminen



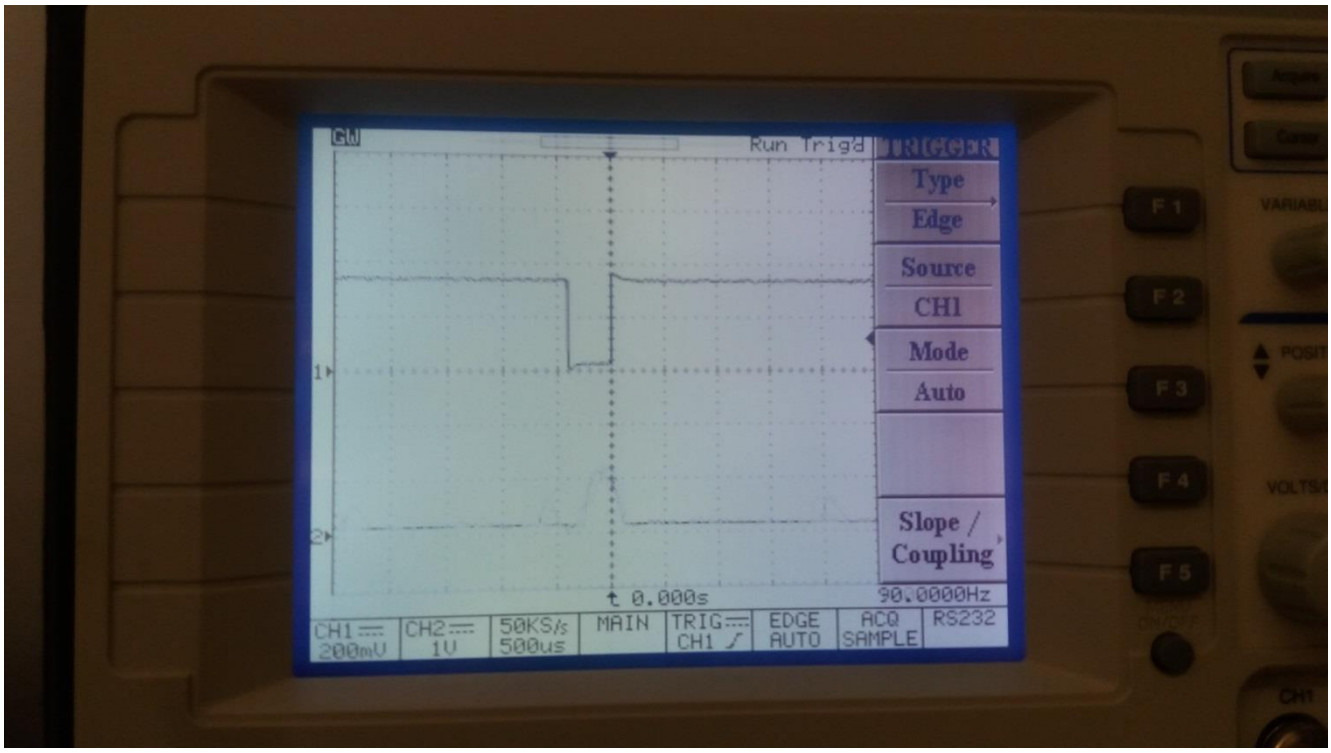
KUVA 20. Ohjaussignaalin tarkastelu oskilloskoopilla

Työssä kytkettiin lähtevän ohjauspulssin kanava ja anturilta tuleva kanava oskilloskoopin, jotta saatiin realistinen kuva, mitä oikeasti tapahtui (KUVA 20). Pulssin luominen tarvitsee ajallisesti tarkkaa ajoitusta sekä pulssin pitää olla PWM-muotoa. PWM-muoto saavutettiin vaihtamalla GPIO-portin tilaa päälle ja pois edellisessä luvussa määritellyin väliajoin. Ohjauspulssi LED:ille syötetään kuvan 17 mukaan kohdasta 6, joka vastaa lattakaapelin reunimmaista kaapelia.

```
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.OUT, initial=GPIO.HIGH)

delay_period=0.50

def readParticleValue():
    arvo3 = 0

    GPIO.output(24, False)# Hamays pulssi, tarvitaan jotta toimii oikein
    time.sleep(0.00032)
    GPIO.output(24, True) #luetaan partikkeli arvo
    time.sleep(delay_period)

    GPIO.output(24, False)
    time.sleep(0.00028) #Pulssin huippu

    arvo3 = adc.read_voltage(1)
    GPIO.output(24, True)
    print("Vs arvo [mV]: %02f" % (arvo3 * 1000))

    return change_mV(arvo3); #Muunnetaan saatu arvo millivoltteihin
```

KUVA 21. Ohjauspulssin luominen V-LED:ille ja sieltä saadun datan luku

Kuvasta 21 huomataan, että ohjauspulssin lähetys ja tulevan arvon luku on tehty metodin sisään. Tämä palauttaa saadun jännitteen millivoltteina. Koodi time.sleep(t) pysäyttää ohjelman toiminnan määräajaksi t, joka ilmoitetaan sekunteina. Aluksi joudumme suorittamaan hämäyspulssin ennen lukevan ohjauspulssin syöttämistä. Ohjaavan pulssin aloitus syötettiin GPIO.output(24, True**)**-komennolla, jonka jälkeen odotettiin kuvan 18 mukaisesti 0.28ms ennen arvon lukemista huipussaan. Lukeminen tapahtui adc.read_voltage(1) -metodilla, josta saatu arvo sijoitettiin muuttujaan arvo3 tulevaa työstöä varten. Kuvan 22 ylemmässä viivassa nähdään ohjauspulssi ja alemmassa anturilta tuleva vastaus.



KUVA 22. Oskilloskooppi ja ohjauspulssi sekä anturilta tuleva vastaus

4.4.3 Ulostulevan signaalin tulkitseminen

Anturilta tuleva on tieto on jännitevaihtelua. Haluttu lopullinen datan muoto on PM2.5 (µg/m³), mutta tätä ulostulevan datan muoto ei anturilta tullessaan vielä ole. Lisäksi ilmassa olevat pölypartikkelit laajenevat, kun ne joutuvat alttiiksi kosteudelle. Lämpökin vaikuttaa mittaustulokseen, mutta käytetyssä tavassa sitä ei tarvitse ottaa huomioon. Vain kosteus siis pitää ottaa huomioon, kun lasketaan partikkelien määrää. Yksinkertaisella kaavalla, joka ottaa myös kosteuden huomioon, voimme arvioida PM2.5-tason. (Sharp corporation 2014, 12.) Saadun datan muuttaminen PM2.5-muotoon, eli Vo → µg/m³ -muunnos sisältää kolme vaihetta. Vaihe 1 on referenssijännitteen hankkiminen, vaihe 2 on muutoksen laskeminen ja vaiheessa 3 arvioidaan PM2.5-taso kaavan avulla.

```
        #Suositeltavaa, etta referessin arvo lasketaan joka kaynnistyken yhteydessa
def calcVs(): #Vs = referenssi arvo, missa vahan partikkeleita. return in mV. Keskiarvollinen tulos

    arvo = 0
    arvo2 = 0

    print("Valmistellaan Vs Referenssi jannitteeen laskemiseen")
    print("Tuuletin sammutetaan 2 min ajaksi")
    turnFanONorOFF(False)
    time.sleep(120)#Odota 2 min, partikkelit laskeuttuu

    for i in range (nayteKerrat):
        GPIO.output(24, False)# Hamays pulssi, tarvitaan jotta toimii oikein
        time.sleep(0.00032)
        GPIO.output(24, True) #luetaan partikkeli arvo
        time.sleep(delay_period)

        GPIO.output(24, False)
        time.sleep(0.00028) #Pulssin huippu

        print("----")
        arvo2 = adc.read_voltage(1)
        print(arvo2)


        arvo += arvo2
        GPIO.output(24, True)
        print("Vs arvo [mV]: %02f" % (arvo2 * 1000))
        print("Vs kokonaisarvo [mV]: %02f" % (arvo * 1000))

    print("_____")

    return (change_mV(arvo) / nayteKerrat);
```

KUVA 23. Referenssijännitteen Vs laskenta

1. Referenssijännitteen Vs hankkiminen

Referenssijännite, eli Vs voidaan saada ympäristöstä, missä on puhdas ilma, kuten puhtaasta muovisesta laatikosta, tai Vs jännite voidaan tallentaa tilassa, missä partikkelit tippuvat painovoiman avulla, kuten kun tuuletin OFF-asennossa. Käytimme jälkimmäistä vaihtoehtoa, eli tuuletinratkaisua. Tuuletinratkaisussa käytettiin apuna luvussa 4.4.4 selitettyä kontrolloitua tuuletinta. Kuvassa 23 Esitetään metodi calcVs(), joka palauttaa keskiarvollisen referenssijännitteen millivoltteina. (Sharp corporation 2014, 12.)

2. Muutoksen laskiminen

ΔV[mV] = (Vo[mV] - Vs[mV]), missä Vs[mV] on referenssijännite millivoltteina ja Vo[mV] on jänniteulostulo, kun tuuletin on päällä. (Sharp corporation 2014, 12.)

```
def checkBeta():

    global Beta
    print("Tarkastetaan ja maaritellaan Beetalle arvo.")

    if(humidity > 50):
        print("humidity > 50")
        Beta = 1-0.01467*(humidity-50)

    elif(humidity <= 50):
        print("humidity <= 50")
        Beta = 1
    else:
        print("error reading humidity")
```

KUVA 24. Betan tarkastus

3. Nyt voimme arvioida PM2.5-tason seuraavalla kaavalla:

   PM2.5 taso (µg/m3) = α × β ×(Vo[mV] – Vs[mV])

   Missä:

   α = Tekijä oikeassa ympäristössä, suositus 0.6

   β = kosteustekijä, määräytyy seuraavasti (h=kosteus prosentteina(%))

   β = {1-0.01467 * (h-50)} jos h > 50

   Tai

   β = 1 jos h ≦ 50

(Sharp corporation 2014, 12.)

Edellä olevassa kaavassa β-symbolin määrittäminen tapahtui kuvan 24 koodin mukaisesti. Kuvassa 25 näytetään, miten PM2.5-arvon laskeminen tapahtui koodissa. Edellä olevat muunnoskaava on löydettävissä valmistajan tarjoamasta dokumentissa (KUVA 26).

```
while True:
    arvo = readParticleValue()

    if(arvo > Vs):
        Tulos = Alpha * Beta * (arvo - Vs)
        print ("Hiukkasen tiheys [ug/m3]: %02f" % (Tulos))
```

KUVA 25. Partikkelin laskenta

Conversion Formula

1. Store a reference voltage (Vs) in the environment with less dust (for example clean box etc). or store a reference voltage (Vs) in the state that after a few minutes to stop the fan (state that dust fell by gravity).
   Note. The output voltage is Vo terminal (pin 2)
2. In case that ⊿ voltage[mV] (Vo[mV] - Vs[mV]) is difference between the reference voltage(Vs) and the output voltage (Vo) when the fan turn on.
   It is possible to approximate the PM2.5 level by use following conversion formula.

Conversion formula (draft):

PM2.5 level (μg/m3) = α × β × (Vo[mV] - Vs[mV])

Note. Do not temperature correction, an estimates in actual environment.

α : Conversion factor in the true environment

Recommendation : 0.6

(β : Humidity factor [h=humidity(%)])

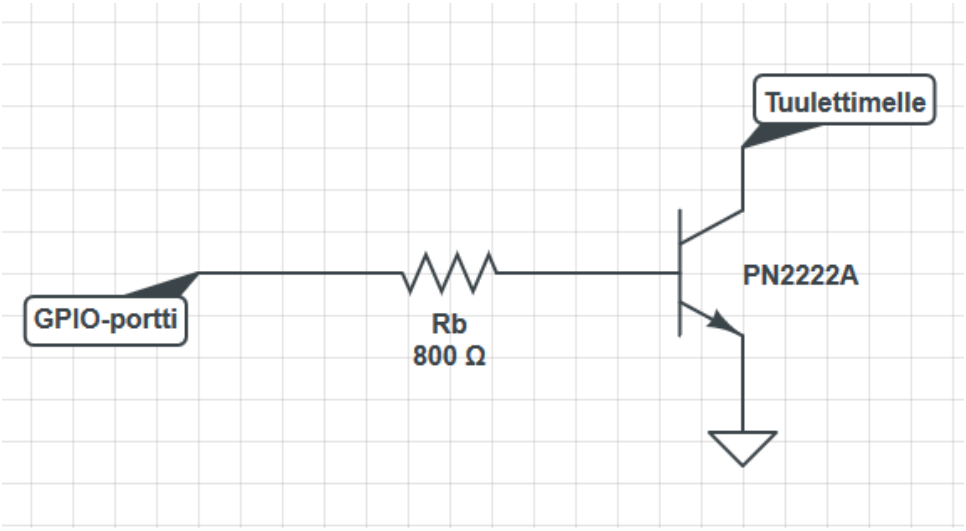
[β = {1-0.01467(h-50)} (h>50)]

[β = 1 (h≦50)]

KUVA 26. Valmistajan kuvaus muunnoksesta (Sharp corporation 2014, 12)

4.4.4 Kontrolloitu tuuletin

Tuli tarpeelliseksi ohjata tuulettimen päälle- ja pois-toimintoa ohjelmallisesti. Kyseistä ominaisuutta tarvittiin, kun laskettiin referenssijännitettä. Kontrolloitavuus saavutettiin käyttämällä GPIO-porttia, transistoria PN2222A ja 800-ohmista etuvastusta.



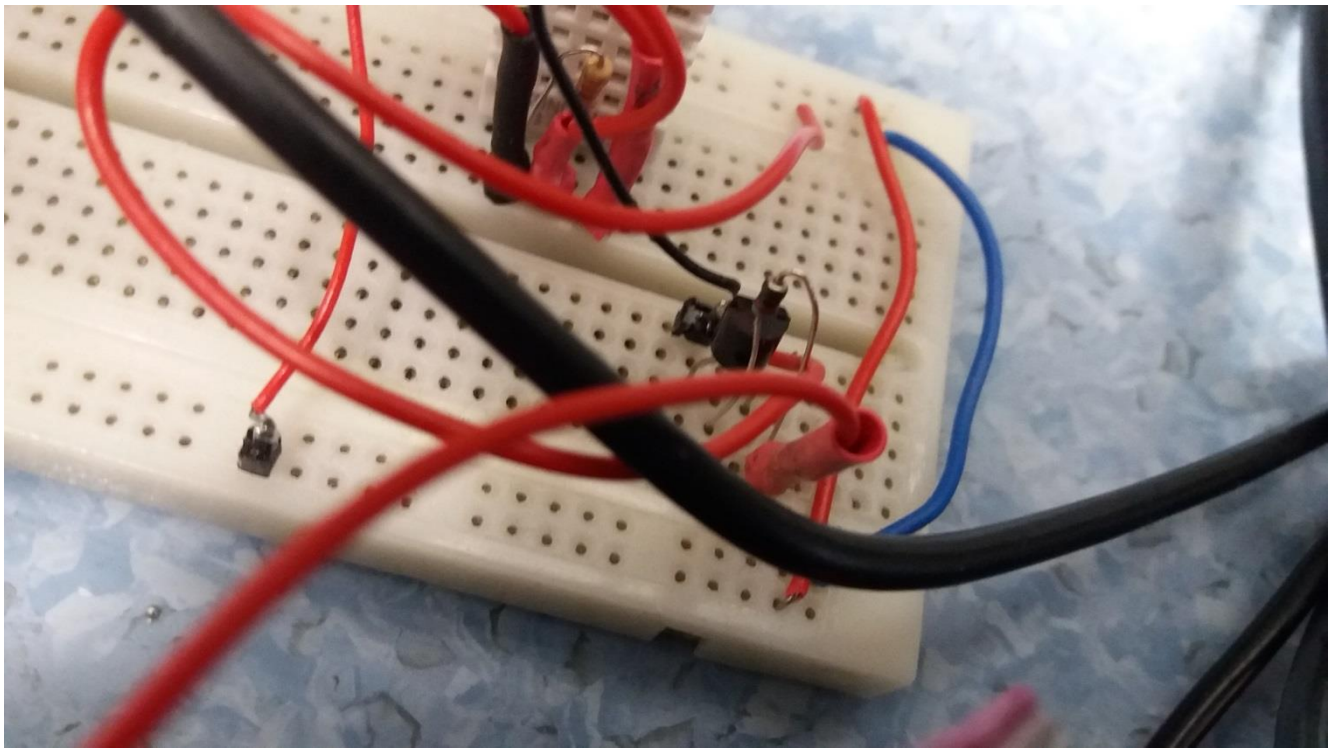
KUVA 27. Tuulettimen kytkentä

Kytkentä tehtiin kuvan 27 mukaisesti. Ohjaus tulee hallinnoimalla GPIO-porttia, ja portiksi valittiin BCM-skeeman mukainen portti 20, jonka eteen laitettiin 800-ohminen etuvastus laskemaan jännitettä PN2222A transistorille. Transistorin kollektorijalka menee tuulettimen jalkaan, josta toinen jalka menee jännitteeseen. Näin saavutetaan kontrolloitavuus GPIO-portin avulla.

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)



def turnFanONorOFF(turnON):

    if(turnON):
        print("Tuuletin kytketty ON")
        GPIO.setup(20, GPIO.OUT)
        GPIO.output(20,True)
    else:
        print("Tuuletin kytketty OFF")
        GPIO.setup(20, GPIO.OUT)
        GPIO.output(20,False)
```

KUVA 28. Tuulettimen kontrollointikoodi

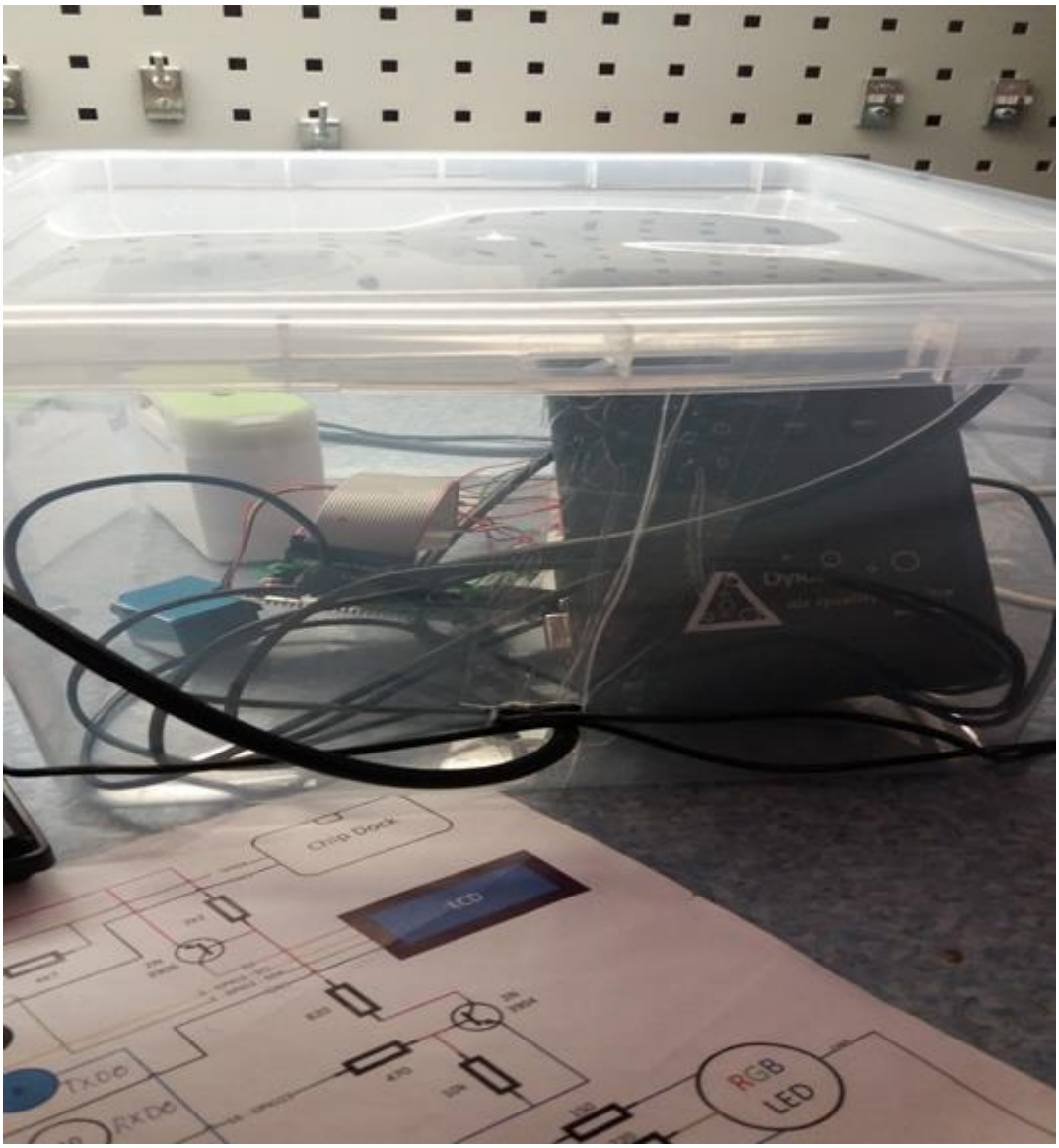
Ohjelmallisesti tuulettimen kontrollointi tapahtui kuvan 29 mukaisesti. Koodi oli yksinkertainen, joten sitä ei käydä lävitse. GPIO-porttien hallinnassa käytettiin BCM-skeemaa ja ylimääräisten virheilmoituksien poistamiseksi käytettiin GPIO.setwarnings(False)-komentoa. Lopputulos tuuletinratkaisusta näytti kuvan 29 mukaiselta.



KUVA 29. Kontrolloidun tuulettimen lopputulos

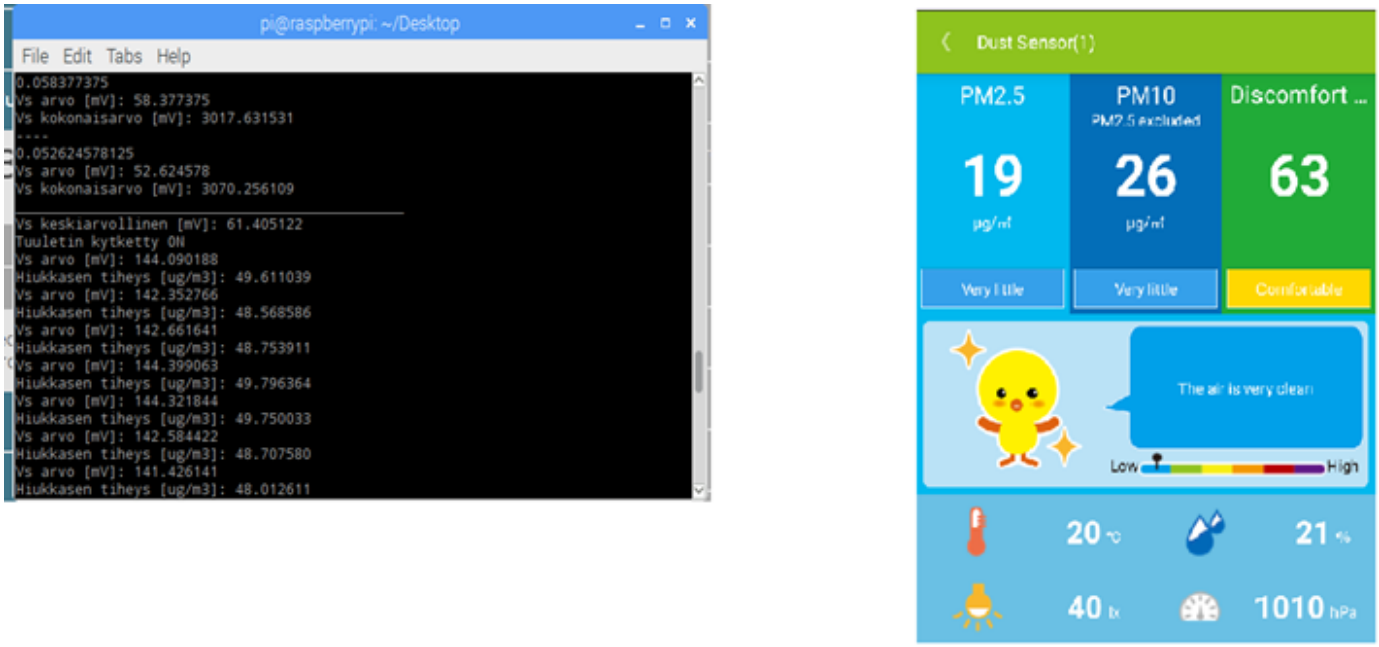
5 TULOSTEN VERTAULU SEKÄ POHDINTA

Saadut tulokset herättivät epäilyksiä, joten oli tarpeen vertailla niitä muihin mittareihin. Vertailussa käytettiin REX-BTPM25-ilmanlaatumittaria sekä Dyloksen valmistamaa dc1100 pro -ilmalaatumittaria. Testauksessa mittauslaitteet laitettiin madollisimman suljettuun tilaan, kuten muoviseen laatikkoon. Kuvassa 30 mittauslaitteet ovat muovilaatikossa valmiina testaukseen.



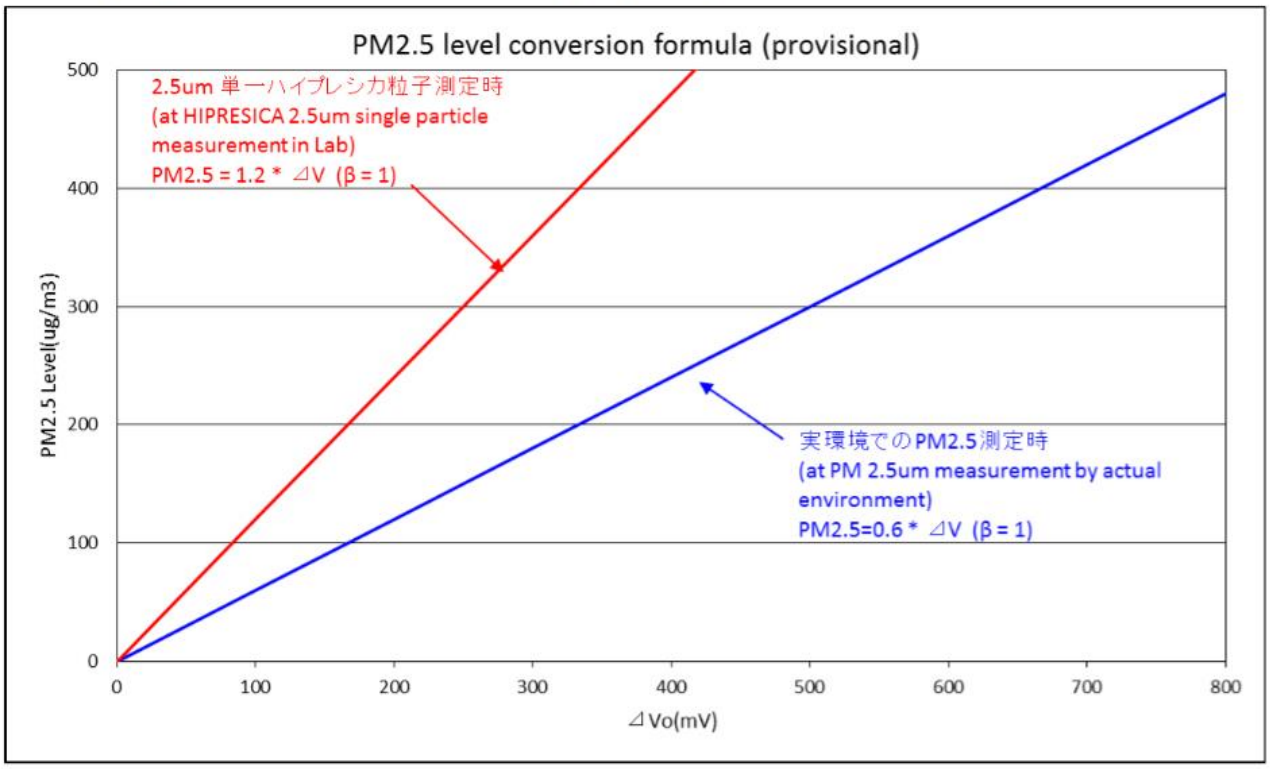
KUVA 30. Testausasema

Laitteilta otettiin näyte mahdollisimman samanaikaisesti. Kuvassa 31 Esitetään mittauslaitteilta saadut tulokset. Kuvassa oikealla on REX-BTPM25-mittarilta saatu tulos ja sekä vasemmalla DN7C3CA006-mittarilta saatu tulos.



KUVA 31. Tulosten vertailu

Kuvasta 31 huomataan, että DN7C3CA006-mittarilla tuleva PM2.5-taso on noin 48 (µg/m3) sekä R EX-BTPM25-mittarilta tuleva PM2.5-taso on 19 (µg/m3). Tuloksien eriarvoisuus herättää epäilyksiä, vaikka DN7C3CA006-mittarilta tuleva arvo on valmistajan tarjoaman kaavion (KUVA 32) mukaan realistinen kyseiseltä anturilta.



KUVA 32. PM2.5 suhteessa ulostuloon Vo (Sharp corporation 2014, 12)

Tuloksena valmistui toimiva kokonaisuus, mutta arvojen oikeellisuuden suhteen olen hieman skeptinen. Huomasin, että olisi ollut hyvä hankkia viereen toinen samanlainen mittari, joka olisi toiminut vertailumittarina. Hankalinta työssä omalta osalta oli PWM-pulssin luominen, koska en ollut ennen kyseistä asiaa tehnyt.

Opinnäytetyön aihe oli todella mielenkiintoinen ja opettavainen. Työtä tehdessä opin paljon lisää paitsi ohjelmoinnista ja Linux-käyttöjärjestelmästä, myös Raspberry Pi:stä ja muista asioista. Aihe vahvisti osaamistani ja antoi suuntaa tulevaisuutta varten.

LÄHTEET

ABElectronics. ADC Pi Plus. Saatavissa: https://www.abelectronics.co.uk/p/56/ADC-Pi-Plus-Raspberry-Pi-Analogue-to-Digital-converter. Viitattu 10.12.2017.

Adafruit. DHT22 temperature-humidity sensor + extras. Saatavissa: https://www.adafruit.com/product/385. Viitattu 11.11.2017

Aosong Electronics. 2017. Digital-output relative humidity & temperature sensor/module AM2303. PDF-dokumentti. Saatavissa: https://cdn-shop.adafruit.com/datasheets/DHT22.pdf. Viitattu 11.12.2017.

Arduino Project Hub. Arduino DTH22 humidity temperature with LCD I2C 16x2 display. Saatavissa: https://create.arduino.cc/projecthub/giftedmedia/arduino-dth22-humidity-temperature-with-lcd-i2c-16x2-display-8fe3c9. Viitattu 1.12.2017

Climate Works. What is House Dust?. Saatavissa: http://www.climateworks.ca/house-dust/. Viitattu 09.10.2017.

Digikey. DN7C3CA006 IR Dust Sensor. Saatavissa: https://www.digikey.com/en/product-highlight/s/sharp-microelectronics/dn7c3ca006-ir-dust-sensor. Viitattu 1.12.2017.

Ensor, D. 2011. Aerosol Science and Technology: History and Reviews. 509. North Carolina: Research Triangle Institute.

Hackster. Air Quality Monitor. Saatavissa: https://www.hackster.io/edwios/air-quality-monitor-3f422f. Viitattu 10.12.1017.

Hiukkastieto. Hiukkasten koko ja muoto. Saatavissa: https://archive.is/gqgew#selection-165.0-165.24. Viitattu 09.12.2017.

Ilmanlaatuportaali. Pienhiukkaset. Saatavissa: http://www.ilmanlaatu.fi/ilmansaasteet/komponentit/pm25.html. Viitattu 10.11.2017.

Jameco. Raspberry Pi Pinout Diagram | Circuit Notes. Saatavissa: https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html. Viitattu 12.12.2017

Sharp corporation. 2014. Device specification for PM2.5 sensor module. PDF-dokumentti. Saatavissa: https://media.digikey.com/pdf/Data%20Sheets/Sharp%20PDFs/DN7C3CA006_Spec.pdf. Viitattu 17.12.2017.

Wikipedia. Raspberry Pi. Saatavissa: https://fi.wikipedia.org/wiki/Raspberry_Pi. Viitattu 11.12.2017.